Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations

Doug Alexander
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Douglass.A.Alexander@jpl.nasa.gov

Payam Zamani, Robert Deen, Paul Andres, Helen Mortensen
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Payam.Zamani@jpl.nasa.gov

Abstract *- During the two years prior and the months subsequent to the historic January, 2004 landing of the Mars Exploration Rover (MER) mission's twin robotic vehicles on the Mars surface, budgetary constraints and growth in mission operations requirements compelled developers of the MER Ground Data System (GDS) at JPL to innovate with robustness at cost-effective levels. One contributing element, the Multimission Image Processing Laboratory (MIPL), was tasked with processing telemetered MER camera data into digital image products necessary for rover traverse planning within a fixed timeline. The design involved systematically transporting, or "pipelining", digital image data between disparate computer processes executed in parallel across multiple machine nodes. The result was an automated system of event-driven product generating systems with sufficient versatility to meet expanding operations needs at affordable costs.*

This paper will discuss, from design to implementation, the methodologies applied to MIPL's automated pipeline processing as a "system of systems" integrated with the MER GDS. Overviews of the interconnected product generating systems will also be provided with emphasis on interdependencies, including those for a) geometric rectification of camera lens distortions, b) generation of stereo disparity, c) derivation of 3-dimensional coordinates in XYZ space, d) generation of unified terrain meshes, e) camera-to-target ranging (distance) and f) multi-image mosaicking. .

Keywords: MER, JPL, MIPL, Mars, rover, image, product, data, pipeline, process, work flow.

1 Introduction

In January of 2004, NASA's Mars Exploration Rover (MER) mission successfully landed the "Spirit" and "Opportunity" rovers on the Mars surface. The techniques involved with remotely operating these mobile vehicles on a distant planet was highly dependent on the ability of mission operations personnel to receive and analyze imaging data that was acquired by each rover's set of engineering and science camera instruments. Once the data were telemetered to the Ground Data System (GDS) at the Jet Propulsion Laboratory (JPL), they had to be processed in such a way so as to optimize the extraction and enhance the quality of navigational information embedded in the images, and with such timeliness that the arduous efforts inherent with analyzing the data and planning same day rover commanding were minimized.

This paper presents a discussion of an automated end-to-end system of data product generating systems designed to accomodate the *in situ* nature of MER rover operations. Developed by JPL's Multimission Image Processing Laboratory (MIPL), the system involved an integration of software programs that processed rover camera instrument data into a variety of unique image data products critical for rover operations traverse planning. Henceforth termed the "Pipeline" for convenience in this paper, the system was equally adept at processing non-image science instrument data for analysis by the science instrument teams. The system's name alludes to the notion that the digital data was sequentially transported, or "pipelined", from one component product generating system to the next. The fundamentals of the Pipeline's event-driven architecture and how they allowed for nearly complete autonomy of the Pipeline operation will be discussed.

The Pipeline's resultant data products were many and their descriptions are extensive. Discussion will touch lightly on the application software developed by MIPL for each data product. Detailing the characteristics of each product type is left as a topic for another paper [1], and instead focus will be placed on discussing the interdependencies between the element application processes as they are laced in the Pipeline's framework.

2 Overview

The basic objective of the Pipeline was to process telemetered camera instrument data into image data products, convert them into terrain maps, and subsequently complete their delivery onto the GDS. The time frame for product delivery had to be short enough to sufficiently allow for planning and uplinking of rover maneuver commands by rover operations short-term planners, the primary customers. Processing within this "tactical" timeline, measured on the order of hours, satisfied the requirements of two other types of operational customers: a) science planners, who were tasked with targeting features of interest found in the images for incorporation into short-term rover traverse plans, and b) mobility analysts, who were responsible for reviewing image data to determine where the rover actually had moved in comparison to the nominal traverse plan for the previous day. A fourth customer, the long-term planners who analyzed multi-image mosaics to plot the course of rover movement several days in advance, operated within a more casual "strategic" timeline measured in days.

Delivery of data products to operational users of the GDS inside JPL's secured flight local area network (LAN) was facilitated by a file server called the Operations Storage Server (OSS). Configured as an immense directory structure hierarchy, the OSS supplanted a customer database on the GDS. Outside the LAN, where dispersed elements of the science instrument teams awaited image data while residing at home institutions, delivery was made using a system called FEI designed at MIPL to provide reliable and secure data transfer across the network. See Section 5.6.3 for more discussion on this system.

3 System Environment

In the MER GDS configuration, the Pipeline was tightly choreographed with a set of upstream processes managed by JPL's System Software (SSW) team and various downstream customer entities. See Figure 1 for a high level diagram of the Pipeline's placement within the context of the MER GDS.

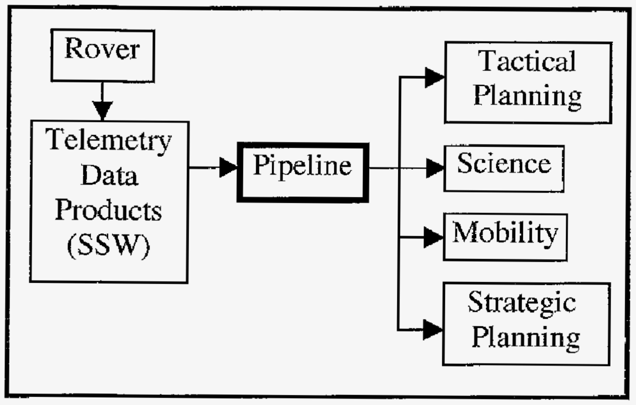

Figure 1. Data Flow in the GDS

3.1 System Hardware

The MER GDS hardware architecture utilized four redundant Sun file servers (NFS) to service a number of Sun/Solaris or Intel-Linux workstations. At the time of initial design, the target platform on the MER GDS for the Pipeline system was unknown. Ultimately, the final Pipeline computing engine was comprised of four dual-processor Intel-Linux workstations per rover mission, each having 1GB of RAM while running at clock speeds of 2.5Mhz.

3.2 Application Software

The Pipeline wouldn't have anything to do if it weren't for the applications that it managed. These were the programs that processed the data into a variety of unique product types. They were in essence the systems that the Pipeline integrated. The Pipeline's architecture allowed application programs to "plug in" with relative ease and minimal configuration. There were several main classes of applications: a) telemetry processing, b) derived image production, c) terrain generation, d) format conversion, e) data delivery, and f) image display.

In addition to the core application programs developed by MIPL, two external programs had to be integrated to support generation of products for the Mini-TES instrument and 3-dimensional terrain meshes.

3.2.1 First Order Products

The MIPL application software supporting the MER project drew a large portion of it's heritage from the Mars Pathfinder (MPF) and Mars Polar Lander (MPL) projects. The telemetry processor ("telemproc") was responsible for digesting raw telemetry data into first order science and operational data products, called Experiment Data Records (EDRs). The telemproc for the all rover engineering and science instruments, with the exception of the Mini-TES instrument, was developed in-house at MIPL and was a direct descendant of the Polar Lander's telemetry processor. The Mini-TES telemproc was developed at Arizona State University.

As shown in Figure 3, the Pipeline processing of EDRs began at the point of interface with SSW processes, which reconstructed the packetized rover instrument telemetry data resident on JPL's Telemetry Data Subsystem (TDS) into data product (DP) file pairs. Comprised of a binary instrument data file and an associated metadata file, each DP was automatically delivered by the SSW processes into an OSS directory called the DP Queue, where they were gathered by the Pipeline for immediate ingestion by the appropriate telemproc.

3.2.2 Derived Products

There were as many as 18 derived image products, called Reduced Data Records (RDR's), generated for each original EDR. A full accounting of each product type is provided elsewhere [1], but the suite included product applications such as radiometric correction, stereo correlation and XYZ generation [2], range (distance) information, robotic arm reachability [3], terrain slope information, and a variety of multi-image mosaic map projections. There were 14 applications written using the "VICAR" image processing system, all based on a common library (Planetary Image Geometry, or PIG) which handled all mission-specific details [4]. They were largely inherited from previous missions such as MPF and MPL and will be further reused in the future Phoenix and Mars Science Laboratory (MSL) missions.

Terrain generation was handled by SUMMITT, a set of terrain building software developed at JPL outside of MIPL. These applications converted the raw XYZ values into a unified terrain mesh used by rover planners for traverse navigation [5].

Figure 2 illustrates the data flow between the various RDR generating processes, starting with the image EDRs.

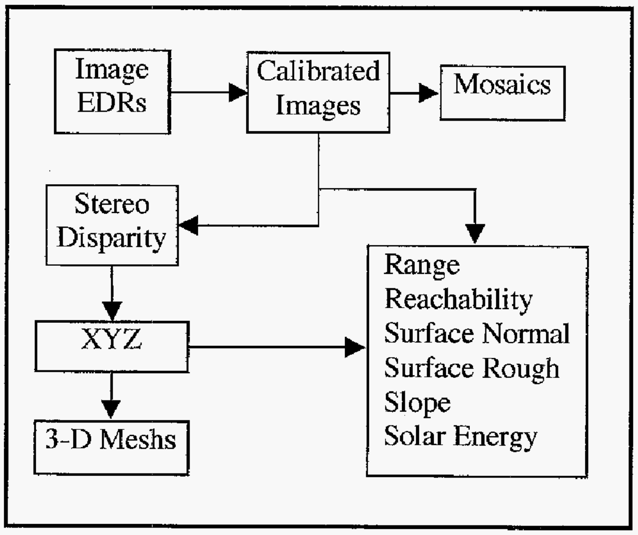

Figure 2. Simplified RDR Application Data Flow

The format conversion application was written in Java using the Image I/O mechanism. It converts any supported format to any other, but was primarily used to convert VICAR-format imagery coming from the RDR generation programs to the standard Planetary Data System (PDS) format required by MER. The important point is that it preserved metadata during the conversion process. It was also used to make JPEG's of the EDR's for public distribution.

A data delivery system (called FEI) and image display system were also written at MIPL, but are generic services used by many missions. See Sections 5.5.3 and 5.5.4, respectively, for details.

Because all of these applications were developed at different sites for different reasons, they were not consistent in terms of calling sequences. Some took simple parameters on the command line, others required input files be constructed. Some required a single image input, some required a stereo pair, and some required a whole collection of inputs. Logging messages were printed and formatted very differently. Most troublesome, the success/fail status returned by the applications were all different. Handling these inconsistencies in the Pipeline turned out to be one of the most challenging aspects of its development.

4 Performance Requirements

There were three major timing requirements levied by the MER project on MIPL for operations and science product generation:

- The requirements called for EDR products to be produced and saved onto the OSS file server within 60 seconds of their arrival on the ground. Actual performance varied from 6 to 12 seconds depending on the instrument and size of data product.
- The requirement for production of RDRs, with the exception of the terrain mesh, was 30 minutes after the end of a telemetry downlink session. During the MER extended mission, new RDRs were introduced, such as solar energy and slope maps [3], and were exempt from meeting this requirement.
- The requirement for generation of the 3-D terrain meshes was one hour from the end of the downlink window. Actual performance varied and occasionally, depending on the size and number of meshes, this requirement was not met for the final mesh. Generation of this product required manual initiation since there was no automated method for broadcasting an end-of-downlink event.

It should be noted that most of the bottleneck in processing was due to the application programs as opposed to the "glueware". Still, the requirements had to be addressed at the time of the Pipeline design so as not to add to the latency already experienced at the application processing level.

In addition to the timing requirements, there were other requirements imposed by MIPL developers for robustness:

- Distribute all products to the MIPL catalog residing outside the flight LAN within 10 seconds of their creation.
- Allow for multiple Pipelines to run in parallel, independent of one another.
- Provide ability to manually reconfigure process loading across multiple workstations. Nominally, four machines were used to support each rover mission's data processing.
- Provide ability to halt or resume execution of the Pipeline at any point in the processing.
- Provide ability to log processing history and have real time tracking of products.
- Provide ability to perform special Pipeline processing "privately" in user directories away from the nominal OSS hierarchy.

5 Design and Implementation

In simple terms, the Pipeline was developed as a single parameterized Bourne shell script that, once initiated by command line at the shell prompt, spawned numerous child process streams across GDS machine resources at the control of the user. Each stream was a serial sequence of specific processes serving a variety of purposes, such as invocation of application software, PDS labeling of data products, and delivery of products to specific directories on the OSS by Sol and instrument type, as well as to external customers outside the LAN.

5.1 Programming Language

Selection of the programming language for the Pipeline development was predicated on a few key factors over a year before the MER mission's landing of the rovers in January of 2004. At that time, the MER project had yet to determine the type of hardware to be used for the GDS, and this prohibited MIPL developers from confidently knowing which versions of various software would be available. While the hardware resources were as yet unknown, it was established that the MER GDS would provide for a Unix-based environment. Since Unix is prevalent on a wide range of computing systems, the probability was high that the typical system user would have some level of Unix experience. The power inherent in the Unix language combined with user familiarity became a prime reason for MIPL developers to build the Pipeline as a Unix shell script. The selection was validated when the MER project settled on Sun and Linux machines as the GDS hardware of choice, with Unix running on both the Solaris and Red Hat operating systems (O/S's), respectively.

The important point is that Unix shell programming languages are known entities and by scripting the Pipeline under Unix shell, the groundwork was laid for easy development of software tools that could supplement or hook into the Pipeline during MER mission operations. And since Unix shells ran on both the Solaris and Linux O/S's of the GDS, deployment of the Pipeline was expanded to multiple user environments.

The Bourne shell was chosen over Perl for a couple of reasons: 1) the version of Perl available on the GDS was incompatible with the version compiled on the MIPL development system, 2) it was felt that scripting in Bourne shell maintained the largest common denominator across the collective knowledge base of the Pipeline developers and operators, and 3) heritage from MPL, wherein the data product generation system was developed under the Bourne-again shell (Bash). It's not to say that selecting Perl as the programming language wouldn't have had its merits as well.

5.2 Constraints

MER project policies governing the GDS constrained the Pipeline in two areas of development. One issue was the regulation that no Data Base Management System such as Sybase, PostgreSQL or MySQL be allowed in the critical path of MER mission operations, and the Pipeline was part of that critical path. So instead of designing a system that utilized a database for operational functions such as auto-triggering of file I/O between processes, an area of design very familiar to MIPL developers as demonstrated during past missions such as MPF and MPL, an alternative strategy had to be adopted. The second issue was a project policy that essentially prevented Pipeline access to a Web server inside the flight LAN, which restrained the distribution of the data product tracking capability.

The resolutions to these issues within the Pipeline design are discussed in subsequent sections in this paper.

5.3 Fundamental Strategy

The fundamental attribute of the design was the ability for each process within a stream to be event-driven. This was exemplified in two general forms within the Pipeline: 1) testing file residence in temporary OSS directories, and 2) testing file attributes by application criteria. Returned status of file residence and criteria testing drove automatic selection of subsequent actions (ie., events) regarding the file's handling, and demonstrated event-driven processing at the lowest level of the Pipeline design.

5.3.1 File Residence Testing

Regarding the first form, in lieu of a relational database's event-triggered capability, each process had built into it a series of endless loops specifically calling the Unix programs "ls" and "find" to search directories on the OSS for files. The temporary directories essentially served as queues that harbored the data for subsequent searching, or polling, by other Pipeline processes. There were up to eight types of directory queues:

- Input Queue - Where all pending input files for a particular application program were stored.
- Input Buffer - An intermediate holding bin where the sets of input files unique by SCLK were moved one at a time from the Input Queue for immediate ingestion by the application program.
- Output Buffer - An intermediate holding bin that received the single set of data processed by the application program for subsequent actions by the Pipeline based on file attributes.
- PDS Queue - Received all data from the Output Buffer destined for PDS labeling.
- PDS Buffer - An intermediate hold bin that received one set of data at a time for immediate ingestion by the PDS labeling system.
- Output Queue – Where the final PDS-labeled versions of data products were received, either from the PDS labeling system or from the Output Buffer, depending on the data product.
- FEI Queue – Where data products destined for external delivery outside the LAN were linked.
- JEDI Queue – Where image EDRs destined for image display were linked.

Additionally, there were other temporary directories that supported contingency processing in the case that data products were not generated, or had to be regenerated: 1) a directory for backup of each data product's input file set, and, 2) directories for file links in the case of failed processing.

5.3.2 File Attribute Testing

Regarding the second form, in the cases of "found" files, their characteristics were tested against criteria for acceptance by the application process that resulted in one of two status types: "success" or "failure".

An example was the need for the stereo correlation process to ingest a pair of images, one acquired using the left camera and the other acquired using the right camera. So for any found image, criteria was designed to test for that image's matching partner, and processing of the image would not proceed until it's partner image was found.

5.4 File Softlinking

Because of the breadth of the OSS directory structure and the product delivery requirements imposed on the Pipeline, file manipulation had to be quick and efficient. This was achieved in the Unix environment by using programs "ln -s" and "cp -s" to softlink data files from directory to directory, minimizing the amount of file copying. Also, file softlinking avoided problems with accessing partially-written files, since the O/S provided for softlink creation to be an atomic process.

5.5 Parallelized Approach

Satisfying the data product delivery requirements necessitated a parallelized stream approach, implicit with the concurrent spawning of multiple child process streams at the outset of the Pipeline's invocation.

The parallelism was at the level of product and process types, and was a function of the number of child streams that could be engineered, with degrees of performance realized through user-controlled distribution of streams across available machine resources. A total of five such streams were identified. Not all product types used all streams, but most used at least three : 1) application stream, 2) PDS labeling stream, and 3) product delivery stream.

5.5.1 Application Stream

This process stream housed the command line call to the application program, and endlessly polled the Input Queue directory for any and all qualified input files. The stream would move a single set of input files unique by SCLK into the Input Buffer directory, from where the application program ingested the data. Upon completion of the processing, the stream deposited the resultant data product into the Output Buffer directory. More fine-grained parallelism could have been had with multiple application streams invoked for the same data product type, and will be a topic for the future.

5.5.2 PDS Labeling Stream

The labeling stream was responsible for calling a Java transcoder program to extract a file's metadata and generate a file-appended label that was compliant to PDS standards. The stream endlessly polled the PDS Queue directory for the candidate files, and moved them one at a time into the PDS Buffer for immediate ingestion by the Java transcoder. Upon completion of the processing, the labeled data product was placed into the Output Queue directory.

5.5.3 Product Delivery Streams (2)

For data product delivery onto the OSS, a stream was spawned to endlessly poll the Output Queue directory for the final PDS-labeled versions of the data products. The stream called the Unix program "mv" to reassign the address of a particular product's file pointer, so that each found file was effectively moved to the file server instead of copied. As part of the move of RDR products, the stream incremented the version number in the product's filename as necessary to avoid overwriting versions of the same product already resident on the OSS.

For data products destined for delivery outside the LAN, yet another process stream was launched to endlessly searched the FEI Queue directory. Found files were then ingested by the File Exchange Interface (FEI) system. FEI was developed as a client/server application to transport data from a data center to client sites, utilizing Kerberos authentication for security [7]. Using FEI programs, data products were copied from the LAN to an external server at MIPL for rerouting to other external client sites.

5.5.4 Image Display Stream

This stream endlessly polled the JEDI Queue directory for EDR image product softlinks. If found, the softlinked file was ingested by client software called the Java EDR Display Interface (JEDI) for image display onto a user-specified monitor. This stream was vital to quick visual quality checking of the image EDRs.

5.6 Error Handling and Messaging

As part of each stream, messages and returned error print statements were generated at both the application program level and the Pipeline glueware level into a single logging text file. There was some inconsistency in the manner by which the application programs returned low level error status, so the Pipeline was engineered to auto-categorize application error types into broad themes and issue additional messages to simplify interpretation.

The log file was set in auto-scroll mode on the workstation monitors for visual monitoring, but it's verbosity made for difficulties in readily interpreting the information in real time. Instead, the greater value found in the log file came with the fact that it provided a permanent record which was searchable at a more leisurely pace in times of anamoly investigations.

5.7 Extensibility

The Unix-based scripting approach to the Pipeline design provided for quick "plug-in" of new capabilities that became necessary due to growing requirements during mission operations.

One example of this Pipeline extensibility came during the extended mission, when several new image products were envisioned that would make operations easier. As Spirit climbed into the Columbia Hills and Opportunity descended into Endurance Crater, the long-term planners realized they needed to be able to visualize the local slope around each respective rover [3]. Power constraints and dusty solar panels led to a need for a product showing locations where solar energy would be maximized. Spirit's ailing right front wheel motivated a product showing climb/descent. All of these were easily implemented using combinations of existing or sightly modified applications.

As another example, the science team used a hybrid version of the Pipeline specially developed to create photometry cubes [6]. This entailed adding yet another incompatible type of application – in this case programs written in IDL (Interactive Data Language, from Research Systems) and very different parameters for many of the standard processing steps. These changes were also readily incorporated in a relatively short time.

In both cases, integration of the new capabilities into the Pipeline was fast and easy. Within the Pipeline script, all application processes were spawned by the same function, so it was simply a matter of adding the command line call to the new application program as a new module block in the code.

6 Operation

6.1 Startup/Shutdown

The Pipeline script was invoked in the Unix shell though command line execution by a single human operator. Processing behavior was controlled via specification of command line parameters.

Although the Pipeline was designed to run autonomously for long periods of time (many days), procedurally, there were advantages to managing the processing sessions in smaller increments. The resulting policy was startup/shutdown of the Pipeline once per Sol, which allowed the operator to maintain the size and number of the temporary working files and directories under limits where NFS performance became noticable.

6.2 Product Verification

6.2.1 GUI-based Product Tracking

Given the complexity of the Pipeline processing and the quantity of data passing through on a daily basis, the need for a means to visually track the progress of processing at the product unit level became apparent. A system called "Product Update Tracking Tool" (a.k.a. PUTT) was created that presented the Pipeline pilot with a web page which visually indicated (via color) the completion status of each image data product.

PUTT was implemented using a small C program and a Perl script outside the Pipeline script and allowed for quick assessment of each product's status by: a) denoting the completion state of each pertinent application program in a graphical spreadsheet (GREEN for success, RED for failure), b) in the cases of failures, isolating and extracting the error statements from the application process logs, and c) providing pop-up window viewability into the log snippets for that product.

As the Pipeline started processing a new EDR, the PUTT program created a small XML "token" file into a temporary Sol directory on the OSS. This file was uniquely named using Spacecraft Clock (SCLK) and Spacecraft Identifier (SCID). As the EDR matriculated through various derived product application processes in the Pipeline, the contents of the token file were updated with status information that included: a) the name of the application program, b) the return status of the program's execution, c) the time of program completion, and d) the name of the program's processing log file pertinent to the EDR.

The Perl script, running every few minutes, collected all unique SCLK tokens from the current Sol and created the web page representation of the status information. The token file contents were concatenated into a single XML file and then converted to HTML by passing the XML through an XSLT filter. If errors were indicated in the token file the script created a second web page containing the appropriate section of the processing log file. Because of project constraints limiting web servers on the flight LAN, the HTML files were copied to a remote web server and were then viewable via a web browser.

6.2.2 Text-based Product Tracking

The use of temporary directory queues to collect file softlinks allowed a simple tool to be written that provided insight into the processing. It was not a GUI representation, but textual, developed as a Perl script to count the number of files in each directory and report as a text listing every 10 seconds or so. Optional parameters were added to control the listings by Sol.

6.3 Private Pipeline Mode

Nominally, the Pipeline placed its output data products into the OSS directory structure, where customers would "shop" for standard data products in subdirectories named by Sol, instrument, and product type. However, there often arose the need to create non-standard data products for special purposes at the request of a customer. These non-standard products could not be copied into the OSS as it would affect all the other customers who were expecting standard data. Therefore a "private" mode of the Pipeline was developed to deliver products into user-specified directories without touching the OSS. This was also extremely useful for MIPL analysts to test new processing methods.

7 Conclusions

The real test of any system is how well it performs in an actual operational setting. The Pipeline has been used daily for over 400 Sols on two rovers, processing in excess of 80,500 EDR's and YYYYYY RDR's as of this writing. While there have been anomalies, none have been serious, and we have met our requirements. The Pipeline has proved itself to be robust and flexible, adapting to a changing mission environment.

While the MER pipeline will not change significantly at this point, it is expected that some derivative of it will be used in future missions. Topics to investigate in the future include more fine-grained parallelism, better product tracking, use of a database (possibly optional) to help manage processing queues, more sophisticated data-flow options, and a more modular, plugin-style approach to application integration.

References

[1] R.G. Deen, D.A. Alexander, J.N. Maki, "Mars Image Products: Science Goes Operational", Proceedings of the 8th International Conference on Space Operations (SpaceOps), Montreal, Canada, 2004.

[2] R.G. Deen, J.J. Lorre, "Seeing in Three Dimensions: Correlation and Triangulation of Mars Exploration Rover Imagery", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[3] C. Leger, R.G. Deen, R.G. Bonitz, "Remote Image Analysis for Mars ExplorationRover Mobility and Manipulation Operations", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[4] R.G. Deen, "Cost Savings through Multimission Code Reuse for Mars ImageProducts", Proceedings of 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, CA, 2003.

[5] J.R. Wright, A. Trebi-Ollenu, J. Morrison, "Terrain Modelling for In-Situ Activity Planning and Rehearsal for the Mars Exploration Rovers", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[6] J.M. Soderblom, J.F. Bell, R.E. Arvidson, J.R. Johnson, M.J. Johnson, F.P. Seelos, "Mars Exploration Rover Pancacm Photometric Data QUBs: Definition and Example Uses", Eos Trans. AGU, Vol 85 No 47, 2004.

[7] T. Huang, "Component Architecture: The Software Architecture for Mission Requirements", Proceedings of 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, CA, 2003.

Notification of Clearance

Mail to:

Douglass.A.Alexander@jpl.nasa.gov, marysue.obrien@jpl.nasa.gov

Notification: Send and Return Cancel

The following title has been cleared by the Document Review Services, Section 274, for public release, presentation and/or printing in the open literature:

Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations

The clearance is CL#05-0680. This clearance is issued for the Meeting Paper and is valid for U.S. and foreign release.

1. Please include the following contractual acknowledgment, preferably at the end of the paper.

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Clearance issued by:

Mary Sue O'Brien
Document Review Services
Section 274

This email is your official Notification of Clearance; Document Review Services no longer issues hard copy clearances.

Did You Know?

Caltech, not the author, is the copyright owner of any material produced as part of the author's employment at JPL. The author does not have the power to transfer copyright.

If the publisher of your manuscript requires a transfer of copyright, please contact Document Review, ext. 4-1141, for assistance if you have not already done so. For more information, see Transferring Copyright Ownership, at http://dmie/cgi/doc-gw.pl?DocID=12009 in JPL Rules!

Page Charges and/or Reprints

If there are page charges for publishing your manuscript or costs for obtaining author's reprints, the JPL Library will process the payments for you.

Once you have an order form or pro forma invoice from the publisher, please contact Barbara Amago, ext. 4-3183, or send your request to 111-113. The Library will need the clearance number and transfer of copyright information from you before the page charges can be paid.

HTML ☐

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109-8099
(818) 354-4321

JPL

05 - 0680

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

Authorization for Public Release and Transfer of Copyright

The contribution entitled "**Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations**" by **Doug Alexander, Payam Zamani, Robert Deen, Paul Andres, and Helen Mortensen**, submitted for publication in the proceedings of the **IEEE 2005 International Conference on Systems, Man, and Cybernetics**, has been cleared for public release by the Jet Propulsion Laboratory, California Institute of Technology. The copyright to the Contribution is transferred to: the **Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE")** when the contribution is published by the above-named publisher with the following reservation of rights:

This Contribution was produced by a member (or members) of the Jet Propulsion Laboratory, California Institute of Technology, and is considered a work-for-hire. In accordance with the contract between the California Institute of Technology and the National Aeronautics and Space Administration, the United States Government, and others acting on its behalf, shall have, for Governmental purposes, a royalty-free, nonexclusive, irrevocable, world-wide license to publish, distribute, copy, exhibit and perform the work, in whole or in part, to authorize others to do so, to reproduce the final published and/or electronic form of the Contribution, to include the work on the NASA/JPL Technical Reports Server web site, and to prepare derivative works including, but not limited to, abstracts, lectures, lecture notes, press releases, reviews, textbooks, reprint books, and translations.

I, as an author, am authorized to sign for and on behalf of all authors, and represent that the information regarding this Contribution, as provided in this agreement, is correct.

[illegible] 7/7/05
(Signature) (Date)

Name: Douglass A. Alexander

AUTHORIZED REPRESENTATIVE

[illegible] 7/7/05
(Signature) (Date)

Name: Adrian Segura

Logistics & Technical Information Division
JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY

IEEE COPYRIGHT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

TITLE OF PAPER/ARTICLE/REPORT/PRESENTATION/SPEECH (hereinafter, "the Work"):

Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations

COMPLETE LIST OF AUTHORS:

Doug Alexander, Payam Zamani, Robert Deen, Paul Andres, Helen Mortensen

IEEE PUBLICATION TITLE (Journal, Magazine, Conference, Book):

International Conference on Systems, Man and Cybernetics 2005

Copyright Transfer

The undersigned hereby assigns to the Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to the above Work, and any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work. The undersigned hereby warrants that the Work is original and that he/she is the author of the Work; to the extent the Work incorporates text passages, figures, data or other material from the works of others, the undersigned has obtained any necessary permissions. **See reverse side for Retained Rights and other Terms and Conditions.**

Author Responsibilities

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements of IEEE Policy 6.4, including provisions covering originality, authorship, author responsibilities and author misconduct. The full policy may be viewed at http://www.ieee.org/about/whatis/policies/p6-4.xml. Authors are advised especially of IEEE Policy 6.4.1B(k): "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE Policy 6.3.B: "It shall be acknowledged that statements and opinions given in work published by the IEEE are the expression of the authors. Responsibility for the content of published papers rests upon the authors, not IEEE."

General Terms

- The undersigned represents that he/she has the power and authority to make and execute this assignment.
- The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
- In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall become null and void and all materials embodying the Work submitted to the IEEE will be destroyed.
- For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.

(1) — Please see attached —

Author/Authorized Agent for Joint Authors **Date**

U.S. Government Employee Certification (where applicable)

This will certify that all authors of the Work are U.S. government employees and prepared the Work on a subject within the scope of their official duties. As such, the Work is not subject to U.S. copyright protection.

(2)____________________

Authorized Signature **Date**

(Authors who are U.S. government employees should also sign signature line (1) above to enable the IEEE to claim and protect its copyright in international jurisdictions.)

Crown Copyright Certification (where applicable)

This will certify that all authors of the Work are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties. As such, the Work is subject to Crown Copyright and is not assigned to the IEEE as set forth in the first sentence of the Copyright Transfer Section above. The undersigned acknowledges, however, that the IEEE has the right to publish, distribute and reprint the Work in all forms and media.

(3)____________________

Authorized Signature **Date**

(Authors who are British or British Commonwealth Government employees should also sign line (1) above to indicate their acceptance of all terms other than the copyright transfer.)

rev. 121302

IEEE COPYRIGHT FORM *(continued)*

RETAINED RIGHTS/TERMS AND CONDITIONS

1. Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.

2. Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.

3. Authors/employers may make limited distribution of all or portions of the Work prior to publication if they inform the IEEE in advance of the nature and extent of such limited distribution.

4. In the case of a Work performed under a U.S. Government contract or grant, the IEEE recognizes that the U.S. Government has royalty-free permission to reproduce all or portions of the Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract/grant so requires.

5. For all uses not covered by items 2, 3, and 4, authors/employers must request permission from the IEEE Intellectual Property Rights office to reproduce or authorize the reproduction of the Work or material extracted verbatim from the Work, including figures and tables.

6. Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use. The IEEE Intellectual Property Rights office must handle all such third-party requests.

INFORMATION FOR AUTHORS

IEEE Copyright Ownership

It is the formal policy of the IEEE to own the copyrights to all copyrightable material in its technical publications and to the individual contributions contained therein, in order to protect the interests of the IEEE, its authors and their employers, and, at the same time, to facilitate the appropriate re-use of this material by others. The IEEE distributes its technical publications throughout the world and does so by various means such as hard copy, microfiche, microfilm, and electronic media. It also abstracts and may translate its publications, and articles contained therein, for inclusion in various compendiums, collective works, databases and similar publications.

Author/Employer Rights

If you are employed and prepared the Work on a subject within the scope of your employment, the copyright in the Work belongs to your employer as a work-for-hire. In that case, the IEEE assumes that when you sign this Form, you are authorized to do so by your employer and that your employer has consented to the transfer of copyright, to the representation and warranty of publication rights, and to all other terms and conditions of this Form. If such authorization and consent has not been given to you, an authorized representative of your employer should sign this Form as the Author.

Reprint/Republication Policy

The IEEE requires that the consent of the first-named author and employer be sought as a condition to granting reprint or republication rights to others or for permitting use of a Work for promotion or marketing purposes.

PLEASE DIRECT ALL QUESTIONS ABOUT THIS FORM TO:
Manager, IEEE Intellectual Property Rights Office, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.
Telephone +1 (732) 562-3966

JPL AUTHORIZATION FOR THE EXTERNAL RELEASE OF INFORMATION

43486

Submit web-site URL or two copies of document with this form to Document Review, 111-207, or email them to docrev@jpl.nasa.gov.

CL No. 05-680
(for DRS use only)

LEAD JPL AUTHOR Douglass A. Alexander	MAIL STOP 168-319	EXTENSION 44316

Approval is required for all JPL scientific and technical information intended for unrestricted external release via print or electronic media. See explanations on page 3 of this form and the Distribute Knowledge documents available through http://dmie.

☐ Original ☐ Modified

I. DOCUMENT AND PROJECT IDENTIFICATION – To be completed by Author/Originator

☐ ABSTRACT (for publication)
☒ FULL PAPER (including poster, video, CD-ROM)
☐ WEB SITE
☐ OTHER ____________
☐ ORAL PRESENTATION
☐ Abstract ☒ Full Text

TITLE
Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations

OTHER AUTHORS
Payam Zamani, Robert Deen, Paul Andres, Helen Mortensen

☐ Premeeting publication
☐ Publication on meeting day
☐ Postmeeting publication
☐ Poster session
☐ Handouts

KEY WORDS FOR INDEXING *(Separate items with commas)*
MER, JPL, MIPL, Mars Exploration Rover, image, product, pipeline

THIS WORK:
☐ Describes technology reported in New Technology Report (NTR) No. ____________
☐ Provides more information for NTR No(s). ____________
☐ Describes **only** science results, data, or theoretical discussions

Publications that describe technology (including software) require an NTR **prior** to clearance. For assistance, contact the Strategic Intellectual Assets Management Office, ext. 3-3421.

LEAD JPL AUTHOR'S SIGNATURE — DATE 3/14/2005

SECTION OR PROJECT LEVEL MANAGER APPROVAL — DATE 3/14/2005
I attest to the quality of information in this material, including its accuracy, relevance and usefulness, audience suitability, clarity, completeness, and lack of bias.

[illegible]

ORIGINATING ORGANIZATION NUMBER *(Section, Project, or Element)*
388

PERFORMING ORGANIZATION *(If different)*

ACCOUNT CODE OR TASK ORDER *(For tracking purposes only)* 102160 03.10.02	DOCUMENT NUMBER(S), RELEASE DATE(S)	DATE RECEIVED 3/14/05	DATE DUE

For presentations, documents, or other scientific/technical information to be externally published (including via electronic media), enter information--such as name, place, and date of conference; periodical or journal name; or book title and publisher--in the area below.

Web Site: Preclearance URL (*JPL internal*) ____________
Postclearance URL (*external*) ____________

☐ Brochure/Newsletter ☐ JPL Publication Section 274 Editor *(If applicable)* ____________
☐ Journal Name ____________
☒ Meeting Title IEEE Conference of Systems, Man and Cybernetics 2005

Meeting Date October 2005 Location Waikoloa, Hawaii, USA
Sponsoring Society IEEE
☐ Book/Book Chapter ☐ Assigned JPL Task ☐ Private Venture Publisher ____________

If your document will not be part of a journal, meeting, or book publication (including a web-based publication), can we post the cleared, final version on the JPL worldwide Technical Report Server (TRS) and send it to the NASA Center for Aerospace Information (CASI)? ☐ Yes ☐ No
(For more information on TRS/CASI, see http://techreports.jpl.nasa.gov and http://www.sti.nasa.gov.)
If your document will be published, the published version will be posted on the TRS and sent to CASI.

II. NATIONAL SECURITY CLASSIFICATION

CHECK ONE (One of the five boxes denoting Security Classification must be checked.)

☐ SECRET ☐ SECRET RD ☐ CONFIDENTIAL ☐ CONFIDENTIAL RD ☒ UNCLASSIFIED

III. AVAILABILITY CATEGORY – To be completed by Document Review

NASA EXPORT-CONTROLLED PROGRAM STI
☐ International Traffic in Arms Regulations (ITAR)
☐ Export Administration Regulations (EAR)

Export-Controlled Document – U.S. Munitions List (USML Category) ____________ or Export Control Classification Number (ECCN) ____________ from the Commerce Control List (CCL) ____________

CONFIDENTAL COMMERCIAL STI
(Check appropriate box below and indicate the distribution limitation if applicable.)
☐ TRADE SECRET ☐ Limited until (date) ____________
☐ SBIR ☐ Limited until (date) ____________
☐ COPYRIGHTED ☐ Limited until (date) ____________
☐ COPYRIGHT TRANSFERRED TO: ☐ Publicly available *(but subject to copying restrictions)* ____________

ADDITIONAL INFORMATION
(Check appropriate distribution limitation below and/or limited until [date], if applicable.)
☐ U.S. Government agencies and U.S. Government agency contractors only
☐ NASA contractors and U.S. Government only ☐ U.S. Government agencies only
☐ NASA personnel and NASA contractors only
☐ Available only with the approval of issuing office ☐ NASA personnel only

III. AVAILABILITY CATEGORY (cont.) – To be completed by Document Review

☒ PUBLICLY AVAILABLE STI	Publicly available means it is unlimited and unclassified, is not export-controlled, does not contain confidential commercial data, and has cleared any applicable patent application.

IV. DOCUMENT DISCLOSING AN INVENTION (For SIAMO Use Only) ROUTED ON

☐ If STI discloses an invention Check box and send to SIAMO	COMMENTS
THIS DOCUMENT MAY BE RELEASED ON *(date)*	STRATEGIC INTELLECTUAL ASSETS MANAGEMENT OFFICE (SIAMO) SIGNATURE DATE

V. BLANKET AVAILABILITY AUTHORIZATION (Optional)

☐ All documents issued under the following contract/grant/project number may be processed as checked in Sections II and III.
This blanket availability authorization is granted on *(date)*________ Check one: ☐ Contract ☐ Grant ☐ Project Number ________

The blanket release authorization granted on *(date)* ________

☐ is RESCINDED – Future documents must have individual availability authorizations.

☐ is MODIFIED – Limitations for all documents processed in the STI system under the blanket release should be changed to conform to blocks as checked in Sections II and III.

SIGNATURE MAIL STOP DATE

VI. PROJECT OFFICER/TECHNICAL MONITOR/DIVISION CHIEF REVIEW OF I THROUGH V

☐ Approval for distributionas marked above ☐ Not approved

NAME OF PROJECT OFFICER OR TECH. MONITOR MAIL STOP SIGNATURE DATE

VII. EXPORT CONTROL REVIEW/CONFIRMATION ROUTED ON

☐ Public release is approved ☐ Public release not approved due to export control ☐ Export-controlled limitation is not applicable

☐ Export-controlled limitation is approved ☐ Export-controlled limitation (ITAR/EAR marked in Section III is assigned to this document)

USML CATEGORY NUMBER (ITAR)	CCL NUMBER, ECCN NUMBER (EAR)	JPL EXPORT CONTROL ADMIN. REPRESENTATIVE SIGNATURE DATE

COMMENTS

VIII. OTHER APPROVALS ROUTED ON

☐ LAUNCH APPROVAL
☐ OFFICE OF COMMUNICATIONS AND EDUCATION
☐ GENERAL COUNSEL
- ☐ Budgetary/Cost Data
- ☐ Vendor Data
- ☐ Copyrights
- ☐ Other ________

☐ OTHER ________

COMMENTS

SIGNATURE DATE

IX. FINAL VERIFICATION, APPROVAL, AND DISPOSITION BY DOCUMENT REVIEW

I have determined that this publication:

☐ DOES contain ITAR/export-controlled, confidential commercial information, and/or discloses an invention and the appropriate limitation is checked in Sections III and/or IV.

☒ Does NOT contain ITAR/export-controlled, confidential commercial information, nor does it disclose an invention and may be released as indicated above.

USML CATEGORY NUMBER (ITAR) ________ CCL NUMBER, ECCN NUMBER (EAR) ________

☒ Public release is approved for U.S. and foreign distribution ☐ Public release is not approved

COMMENTS Add NASA acknowledgment

SIGNATURE Mary Sue O'Brien MAIL STOP DATE 3/14/05

☐ Obtained published version Date ________ ☐ Obtained final JPL version Date ________

See page 3 for instructions for completing this form.

Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations

Doug Alexander
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Douglass.A.Alexander@jpl.nasa.gov

Payam Zamani, Robert Deen, Paul Andres, Helen Mortensen
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Payam.Zamani@jpl.nasa.gov

Abstract *- During the two years prior and the months subsequent to the historic January, 2004 landing of the Mars Exploration Rover (MER) mission's twin robotic vehicles on the Mars surface, budgetary constraints and growth in mission operations requirements compelled developers of the MER Ground Data System (GDS) at JPL to innovate with robustness at cost-effective levels. One contributing element, the Multimission Image Processing Laboratory (MIPL), was tasked with processing telemetered MER camera data into digital image products necessary for rover traverse planning within a fixed timeline. The design involved systematically transporting, or "pipelining", digital image data between disparate computer processes executed in parallel across multiple machine nodes. The result was an automated system of event-driven product generating systems with sufficient versatility to meet expanding operations needs at affordable costs.*

This paper will discuss, from design to implementation, the methodologies applied to MIPL's automated pipeline processing as a "system of systems" integrated with the MER GDS. Overviews of the interconnected product generating systems will also be provided with emphasis on interdependencies, including those for a) geometric rectification of camera lens distortions, b) generation of stereo disparity, c) derivation of 3-dimensional coordinates in XYZ space, d) generation of unified terrain meshes, e) camera-to-target ranging (distance) and f) multi-image mosaicking. .

Keywords: MER, JPL, MIPL, Mars, rover, image, product, data, pipeline, process, work flow.

1 Introduction

In January of 2004, NASA's Mars Exploration Rover (MER) mission successfully landed the "Spirit" and "Opportunity" rovers on the Mars surface. The techniques involved with remotely operating these mobile vehicles on a distant planet was highly dependent on the ability of mission operations personnel to receive and analyze imaging data that was acquired by each rover's set of engineering and science camera instruments. Once the data were telemetered to the Ground Data System (GDS) at the Jet Propulsion Laboratory (JPL), they had to be processed in such a way so as to optimize the extraction and enhance the quality of navigational information embedded in the images, and with such timeliness that the arduous efforts inherent with analyzing the data and planning same day rover commanding were minimized.

This paper presents a discussion of an automated end-to-end system of data product generating systems designed to accomodate the *in situ* nature of MER rover operations. Developed by JPL's Multimission Image Processing Laboratory (MIPL), the system involved an integration of software programs that processed rover camera instrument data into a variety of unique image data products critical for rover operations traverse planning. Henceforth termed the "Pipeline" for convenience in this paper, the system was equally adept at processing non-image science instrument data for analysis by the science instrument teams. The system's name alludes to the notion that the digital data was sequentially transported, or "pipelined", from one component product generating system to the next. The fundamentals of the Pipeline's event-driven architecture and how they allowed for nearly complete autonomy of the Pipeline operation will be discussed.

The Pipeline's resultant data products were many and their descriptions are extensive. Discussion will touch lightly on the application software developed by MIPL for each data product. Detailing the characteristics of each product type is left as a topic for another paper [1], and instead focus will be placed on discussing the interdependencies between the element application processes as they are laced in the Pipeline's framework.

2 Overview

The basic objective of the Pipeline was to process telemetered camera instrument data into image data products, convert them into terrain maps, and subsequently complete their delivery onto the GDS. The time frame for product delivery had to be short enough to sufficiently allow for planning and uplinking of rover maneuver commands by rover operations short-term planners, the primary customers. Processing within this "tactical" timeline, measured on the order of hours, satisfied the requirements of two other types of operational customers: a) science planners, who were tasked with targeting features of interest found in the images for incorporation into short-term rover traverse plans, and b) mobility analysts, who were responsible for reviewing image data to determine where the rover actually had moved in comparison to the nominal traverse plan for the previous day. A fourth customer, the long-term planners who analyzed multi-image mosaics to plot the course of rover movement several days in advance, operated within a more casual "strategic" timeline measured in days.

Delivery of data products to operational users of the GDS inside JPL's secured flight local area network (LAN) was facilitated by a file server called the Operations Storage Server (OSS). Configured as an immense directory structure hierarchy, the OSS supplanted a customer database on the GDS. Outside the LAN, where dispersed elements of the science instrument teams awaited image data while residing at home institutions, delivery was made using a system called FEI designed at MIPL to provide reliable and secure data transfer across the network. See Section 5.6.3 for more discussion on this system.

3 System Environment

In the MER GDS configuration, the Pipeline was tightly choreographed with a set of upstream processes managed by JPL's System Software (SSW) team and various downstream customer entities. See Figure 1 for a high level diagram of the Pipeline's placement within the context of the MER GDS.

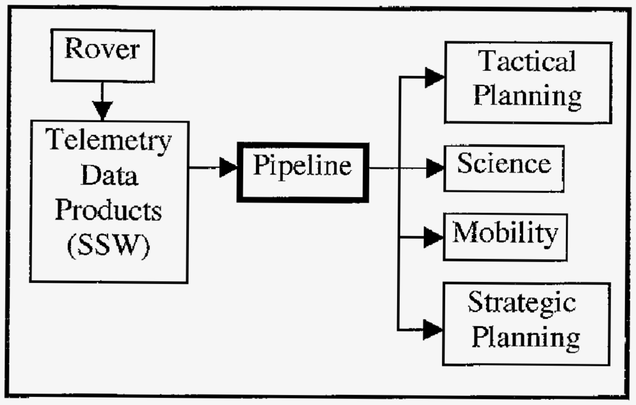

Figure 1. Data Flow in the GDS

3.1 System Hardware

The MER GDS hardware architecture utilized four redundant Sun file servers (NFS) to service a number of Sun/Solaris or Intel-Linux workstations. At the time of initial design, the target platform on the MER GDS for the Pipeline system was unknown. Ultimately, the final Pipeline computing engine was comprised of four dual-processor Intel-Linux workstations per rover mission, each having 1GB of RAM while running at clock speeds of 2.5Mhz.

3.2 Application Software

The Pipeline wouldn't have anything to do if it weren't for the applications that it managed. These were the programs that processed the data into a variety of unique product types. They were in essence the systems that the Pipeline integrated. The Pipeline's architecture allowed application programs to "plug in" with relative ease and minimal configuration. There were several main classes of applications: a) telemetry processing, b) derived image production, c) terrain generation, d) format conversion, e) data delivery, and f) image display.

In addition to the core application programs developed by MIPL, two external programs had to be integrated to support generation of products for the Mini-TES instrument and 3-dimensional terrain meshes.

3.2.1 First Order Products

The MIPL application software supporting the MER project drew a large portion of it's heritage from the Mars Pathfinder (MPF) and Mars Polar Lander (MPL) projects. The telemetry processor ("telemproc") was responsible for digesting raw telemetry data into first order science and operational data products, called Experiment Data Records (EDRs). The telemproc for the all rover engineering and science instruments, with the exception of the Mini-TES instrument, was developed in-house at MIPL and was a direct descendant of the Polar Lander's telemetry processor. The Mini-TES telemproc was developed at Arizona State University.

As shown in Figure 3, the Pipeline processing of EDRs began at the point of interface with SSW processes, which reconstructed the packetized rover instrument telemetry data resident on JPL's Telemetry Data Subsystem (TDS) into data product (DP) file pairs. Comprised of a binary instrument data file and an associated metadata file, each DP was automatically delivered by the SSW processes into an OSS directory called the DP Queue, where they were gathered by the Pipeline for immediate ingestion by the appropriate telemproc.

3.2.2 Derived Products

There were as many as 18 derived image products, called Reduced Data Records (RDR's), generated for each original EDR. A full accounting of each product type is provided elsewhere [1], but the suite included product applications such as radiometric correction, stereo correlation and XYZ generation [2], range (distance) information, robotic arm reachability [3], terrain slope information, and a variety of multi-image mosaic map projections. There were 14 applications written using the "VICAR" image processing system, all based on a common library (Planetary Image Geometry, or PIG) which handled all mission-specific details [4]. They were largely inherited from previous missions such as MPF and MPL and will be further reused in the future Phoenix and Mars Science Laboratory (MSL) missions.

Terrain generation was handled by SUMMITT, a set of terrain building software developed at JPL outside of MIPL. These applications converted the raw XYZ values into a unified terrain mesh used by rover planners for traverse navigation [5].

Figure 2 illustrates the data flow between the various RDR generating processes, starting with the image EDRs.

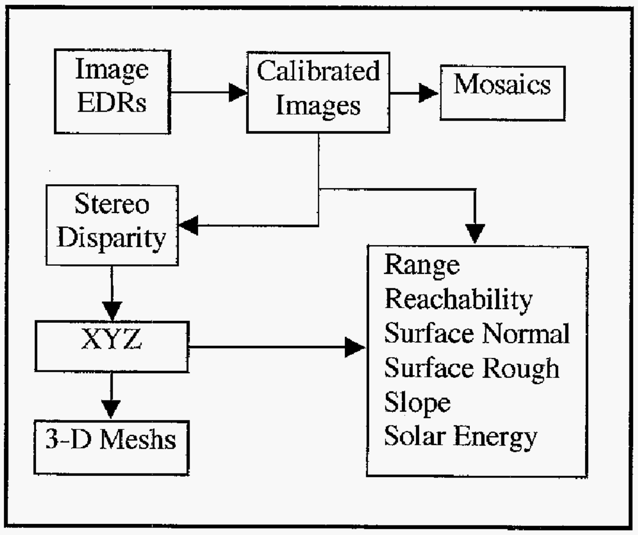

Figure 2. Simplified RDR Application Data Flow

The format conversion application was written in Java using the Image I/O mechanism. It converts any supported format to any other, but was primarily used to convert VICAR-format imagery coming from the RDR generation programs to the standard Planetary Data System (PDS) format required by MER. The important point is that it preserved metadata during the conversion process. It was also used to make JPEG's of the EDR's for public distribution.

A data delivery system (called FEI) and image display system were also written at MIPL, but are generic services used by many missions. See Sections 5.5.3 and 5.5.4, respectively, for details.

Because all of these applications were developed at different sites for different reasons, they were not consistent in terms of calling sequences. Some took simple parameters on the command line, others required input files be constructed. Some required a single image input, some required a stereo pair, and some required a whole collection of inputs. Logging messages were printed and formatted very differently. Most troublesome, the success/fail status returned by the applications were all different. Handling these inconsistencies in the Pipeline turned out to be one of the most challenging aspects of its development.

4 Performance Requirements

There were three major timing requirements levied by the MER project on MIPL for operations and science product generation:

- The requirements called for EDR products to be produced and saved onto the OSS file server within 60 seconds of their arrival on the ground. Actual performance varied from 6 to 12 seconds depending on the instrument and size of data product.
- The requirement for production of RDRs, with the exception of the terrain mesh, was 30 minutes after the end of a telemetry downlink session. During the MER extended mission, new RDRs were introduced, such as solar energy and slope maps [3], and were exempt from meeting this requirement.
- The requirement for generation of the 3-D terrain meshes was one hour from the end of the downlink window. Actual performance varied and occasionally, depending on the size and number of meshes, this requirement was not met for the final mesh. Generation of this product required manual initiation since there was no automated method for broadcasting an end-of-downlink event.

It should be noted that most of the bottleneck in processing was due to the application programs as opposed to the "glueware". Still, the requirements had to be addressed at the time of the Pipeline design so as not to add to the latency already experienced at the application processing level.

In addition to the timing requirements, there were other requirements imposed by MIPL developers for robustness:

- Distribute all products to the MIPL catalog residing outside the flight LAN within 10 seconds of their creation.
- Allow for multiple Pipelines to run in parallel, independent of one another.
- Provide ability to manually reconfigure process loading across multiple workstations. Nominally, four machines were used to support each rover mission's data processing.
- Provide ability to halt or resume execution of the Pipeline at any point in the processing.
- Provide ability to log processing history and have real time tracking of products.
- Provide ability to perform special Pipeline processing "privately" in user directories away from the nominal OSS hierarchy.

5 Design and Implementation

In simple terms, the Pipeline was developed as a single parameterized Bourne shell script that, once initiated by command line at the shell prompt, spawned numerous child process streams across GDS machine resources at the control of the user. Each stream was a serial sequence of specific processes serving a variety of purposes, such as invocation of application software, PDS labeling of data products, and delivery of products to specific directories on the OSS by Sol and instrument type, as well as to external customers outside the LAN.

5.1 Programming Language

Selection of the programming language for the Pipeline development was predicated on a few key factors over a year before the MER mission's landing of the rovers in January of 2004. At that time, the MER project had yet to determine the type of hardware to be used for the GDS, and this prohibited MIPL developers from confidently knowing which versions of various software would be available. While the hardware resources were as yet unknown, it was established that the MER GDS would provide for a Unix-based environment. Since Unix is prevalent on a wide range of computing systems, the probability was high that the typical system user would have some level of Unix experience. The power inherent in the Unix language combined with user familiarity became a prime reason for MIPL developers to build the Pipeline as a Unix shell script. The selection was validated when the MER project settled on Sun and Linux machines as the GDS hardware of choice, with Unix running on both the Solaris and Red Hat operating systems (O/S's), respectively.

The important point is that Unix shell programming languages are known entities and by scripting the Pipeline under Unix shell, the groundwork was laid for easy development of software tools that could supplement or hook into the Pipeline during MER mission operations. And since Unix shells ran on both the Solaris and Linux O/S's of the GDS, deployment of the Pipeline was expanded to multiple user environments.

The Bourne shell was chosen over Perl for a couple of reasons: 1) the version of Perl available on the GDS was incompatible with the version compiled on the MIPL development system, 2) it was felt that scripting in Bourne shell maintained the largest common denominator across the collective knowledge base of the Pipeline developers and operators, and 3) heritage from MPL, wherein the data product generation system was developed under the Bourne-again shell (Bash). It's not to say that selecting Perl as the programming language wouldn't have had its merits as well.

5.2 Constraints

MER project policies governing the GDS constrained the Pipeline in two areas of development. One issue was the regulation that no Data Base Management System such as Sybase, PostgreSQL or MySQL be allowed in the critical path of MER mission operations, and the Pipeline was part of that critical path. So instead of designing a system that utilized a database for operational functions such as auto-triggering of file I/O between processes, an area of design very familiar to MIPL developers as demonstrated during past missions such as MPF and MPL, an alternative strategy had to be adopted. The second issue was a project policy that essentially prevented Pipeline access to a Web server inside the flight LAN, which restrained the distribution of the data product tracking capability.

The resolutions to these issues within the Pipeline design are discussed in subsequent sections in this paper.

5.3 Fundamental Strategy

The fundamental attribute of the design was the ability for each process within a stream to be event-driven. This was exemplified in two general forms within the Pipeline: 1) testing file residence in temporary OSS directories, and 2) testing file attributes by application criteria. Returned status of file residence and criteria testing drove automatic selection of subsequent actions (ie., events) regarding the file's handling, and demonstrated event-driven processing at the lowest level of the Pipeline design.

5.3.1 File Residence Testing

Regarding the first form, in lieu of a relational database's event-triggered capability, each process had built into it a series of endless loops specifically calling the Unix programs "ls" and "find" to search directories on the OSS for files. The temporary directories essentially served as queues that harbored the data for subsequent searching, or polling, by other Pipeline processes. There were up to eight types of directory queues:

- Input Queue - Where all pending input files for a particular application program were stored.
- Input Buffer - An intermediate holding bin where the sets of input files unique by SCLK were moved one at a time from the Input Queue for immediate ingestion by the application program.
- Output Buffer - An intermediate holding bin that received the single set of data processed by the application program for subsequent actions by the Pipeline based on file attributes.
- PDS Queue - Received all data from the Output Buffer destined for PDS labeling.
- PDS Buffer - An intermediate hold bin that received one set of data at a time for immediate ingestion by the PDS labeling system.
- Output Queue – Where the final PDS-labeled versions of data products were received, either from the PDS labeling system or from the Output Buffer, depending on the data product.
- FEI Queue – Where data products destined for external delivery outside the LAN were linked.
- JEDI Queue – Where image EDRs destined for image display were linked.

Additionally, there were other temporary directories that supported contingency processing in the case that data products were not generated, or had to be regenerated: 1) a directory for backup of each data product's input file set, and, 2) directories for file links in the case of failed processing.

5.3.2 File Attribute Testing

Regarding the second form, in the cases of "found" files, their characteristics were tested against criteria for acceptance by the application process that resulted in one of two status types: "success" or "failure".

An example was the need for the stereo correlation process to ingest a pair of images, one acquired using the left camera and the other acquired using the right camera. So for any found image, criteria was designed to test for that image's matching partner, and processing of the image would not proceed until it's partner image was found.

5.4 File Softlinking

Because of the breadth of the OSS directory structure and the product delivery requirements imposed on the Pipeline, file manipulation had to be quick and efficient. This was achieved in the Unix environment by using programs "ln -s" and "cp -s" to softlink data files from directory to directory, minimizing the amount of file copying. Also, file softlinking avoided problems with accessing partially-written files, since the O/S provided for softlink creation to be an atomic process.

5.5 Parallelized Approach

Satisfying the data product delivery requirements necessitated a parallelized stream approach, implicit with the concurrent spawning of multiple child process streams at the outset of the Pipeline's invocation.

The parallelism was at the level of product and process types, and was a function of the number of child streams that could be engineered, with degrees of performance realized through user-controlled distribution of streams across available machine resources. A total of five such streams were identified. Not all product types used all streams, but most used at least three : 1) application stream, 2) PDS labeling stream, and 3) product delivery stream.

5.5.1 Application Stream

This process stream housed the command line call to the application program, and endlessly polled the Input Queue directory for any and all qualified input files. The stream would move a single set of input files unique by SCLK into the Input Buffer directory, from where the application program ingested the data. Upon completion of the processing, the stream deposited the resultant data product into the Output Buffer directory. More fine-grained parallelism could have been had with multiple application streams invoked for the same data product type, and will be a topic for the future.

5.5.2 PDS Labeling Stream

The labeling stream was responsible for calling a Java transcoder program to extract a file's metadata and generate a file-appended label that was compliant to PDS standards. The stream endlessly polled the PDS Queue directory for the candidate files, and moved them one at a time into the PDS Buffer for immediate ingestion by the Java transcoder. Upon completion of the processing, the labeled data product was placed into the Output Queue directory.

5.5.3 Product Delivery Streams (2)

For data product delivery onto the OSS, a stream was spawned to endlessly poll the Output Queue directory for the final PDS-labeled versions of the data products. The stream called the Unix program "mv" to reassign the address of a particular product's file pointer, so that each found file was effectively moved to the file server instead of copied. As part of the move of RDR products, the stream incremented the version number in the product's filename as necessary to avoid overwriting versions of the same product already resident on the OSS.

For data products destined for delivery outside the LAN, yet another process stream was launched to endlessly searched the FEI Queue directory. Found files were then ingested by the File Exchange Interface (FEI) system. FEI was developed as a client/server application to transport data from a data center to client sites, utilizing Kerberos authentication for security [7]. Using FEI programs, data products were copied from the LAN to an external server at MIPL for rerouting to other external client sites.

5.5.4 Image Display Stream

This stream endlessly polled the JEDI Queue directory for EDR image product softlinks. If found, the softlinked file was ingested by client software called the Java EDR Display Interface (JEDI) for image display onto a user-specified monitor. This stream was vital to quick visual quality checking of the image EDRs.

5.6 Error Handling and Messaging

As part of each stream, messages and returned error print statements were generated at both the application program level and the Pipeline glueware level into a single logging text file. There was some inconsistency in the manner by which the application programs returned low level error status, so the Pipeline was engineered to auto-categorize application error types into broad themes and issue additional messages to simplify interpretation.

The log file was set in auto-scroll mode on the workstation monitors for visual monitoring, but it's verbosity made for difficulties in readily interpreting the information in real time. Instead, the greater value found in the log file came with the fact that it provided a permanent record which was searchable at a more leisurely pace in times of anamoly investigations.

5.7 Extensibility

The Unix-based scripting approach to the Pipeline design provided for quick "plug-in" of new capabilities that became necessary due to growing requirements during mission operations.

One example of this Pipeline extensibility came during the extended mission, when several new image products were envisioned that would make operations easier. As Spirit climbed into the Columbia Hills and Opportunity descended into Endurance Crater, the long-term planners realized they needed to be able to visualize the local slope around each respective rover [3]. Power constraints and dusty solar panels led to a need for a product showing locations where solar energy would be maximized. Spirit's ailing right front wheel motivated a product showing climb/descent. All of these were easily implemented using combinations of existing or sightly modified applications.

As another example, the science team used a hybrid version of the Pipeline specially developed to create photometry cubes [6]. This entailed adding yet another incompatible type of application – in this case programs written in IDL (Interactive Data Language, from Research Systems) and very different parameters for many of the standard processing steps. These changes were also readily incorporated in a relatively short time.

In both cases, integration of the new capabilities into the Pipeline was fast and easy. Within the Pipeline script, all application processes were spawned by the same function, so it was simply a matter of adding the command line call to the new application program as a new module block in the code.

6 Operation

6.1 Startup/Shutdown

The Pipeline script was invoked in the Unix shell though command line execution by a single human operator. Processing behavior was controlled via specification of command line parameters.

Although the Pipeline was designed to run autonomously for long periods of time (many days), procedurally, there were advantages to managing the processing sessions in smaller increments. The resulting policy was startup/shutdown of the Pipeline once per Sol, which allowed the operator to maintain the size and number of the temporary working files and directories under limits where NFS performance became noticable.

6.2 Product Verification

6.2.1 GUI-based Product Tracking

Given the complexity of the Pipeline processing and the quantity of data passing through on a daily basis, the need for a means to visually track the progress of processing at the product unit level became apparent. A system called "Product Update Tracking Tool" (a.k.a. PUTT) was created that presented the Pipeline pilot with a web page which visually indicated (via color) the completion status of each image data product.

PUTT was implemented using a small C program and a Perl script outside the Pipeline script and allowed for quick assessment of each product's status by: a) denoting the completion state of each pertinent application program in a graphical spreadsheet (GREEN for success, RED for failure), b) in the cases of failures, isolating and extracting the error statements from the application process logs, and c) providing pop-up window viewability into the log snippets for that product.

As the Pipeline started processing a new EDR, the PUTT program created a small XML "token" file into a temporary Sol directory on the OSS. This file was uniquely named using Spacecraft Clock (SCLK) and Spacecraft Identifier (SCID). As the EDR matriculated through various derived product application processes in the Pipeline, the contents of the token file were updated with status information that included: a) the name of the application program, b) the return status of the program's execution, c) the time of program completion, and d) the name of the program's processing log file pertinent to the EDR.

The Perl script, running every few minutes, collected all unique SCLK tokens from the current Sol and created the web page representation of the status information. The token file contents were concatenated into a single XML file and then converted to HTML by passing the XML through an XSLT filter. If errors were indicated in the token file the script created a second web page containing the appropriate section of the processing log file. Because of project constraints limiting web servers on the flight LAN, the HTML files were copied to a remote web server and were then viewable via a web browser.

6.2.2 Text-based Product Tracking

The use of temporary directory queues to collect file softlinks allowed a simple tool to be written that provided insight into the processing. It was not a GUI representation, but textual, developed as a Perl script to count the number of files in each directory and report as a text listing every 10 seconds or so. Optional parameters were added to control the listings by Sol.

6.3 Private Pipeline Mode

Nominally, the Pipeline placed its output data products into the OSS directory structure, where customers would "shop" for standard data products in subdirectories named by Sol, instrument, and product type. However, there often arose the need to create non-standard data products for special purposes at the request of a customer. These non-standard products could not be copied into the OSS as it would affect all the other customers who were expecting standard data. Therefore a "private" mode of the Pipeline was developed to deliver products into user-specified directories without touching the OSS. This was also extremely useful for MIPL analysts to test new processing methods.

7 Conclusions

The real test of any system is how well it performs in an actual operational setting. The Pipeline has been used daily for over 400 Sols on two rovers, processing in excess of 80,500 EDR's and YYYYYY RDR's as of this writing. While there have been anomalies, none have been serious, and we have met our requirements. The Pipeline has proved itself to be robust and flexible, adapting to a changing mission environment.

While the MER pipeline will not change significantly at this point, it is expected that some derivative of it will be used in future missions. Topics to investigate in the future include more fine-grained parallelism, better product tracking, use of a database (possibly optional) to help manage processing queues, more sophisticated data-flow options, and a more modular, plugin-style approach to application integration.

References

[1] R.G. Deen, D.A. Alexander, J.N. Maki, "Mars Image Products: Science Goes Operational", Proceedings of the 8th International Conference on Space Operations (SpaceOps), Montreal, Canada, 2004.

[2] R.G. Deen, J.J. Lorre, "Seeing in Three Dimensions: Correlation and Triangulation of Mars Exploration Rover Imagery", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[3] C. Leger, R.G. Deen, R.G. Bonitz, "Remote Image Analysis for Mars ExplorationRover Mobility and Manipulation Operations", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[4] R.G. Deen, "Cost Savings through Multimission Code Reuse for Mars ImageProducts", Proceedings of 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, CA, 2003.

[5] J.R. Wright, A. Trebi-Ollenu, J. Morrison, "Terrain Modelling for In-Situ Activity Planning and Rehearsal for the Mars Exploration Rovers", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[6] J.M. Soderblom, J.F. Bell, R.E. Arvidson, J.R. Johnson, M.J. Johnson, F.P. Seelos, "Mars Exploration Rover Pancacm Photometric Data QUBs: Definition and Example Uses", Eos Trans. AGU, Vol 85 No 47, 2004.

[7] T. Huang, "Component Architecture: The Software Architecture for Mission Requirements", Proceedings of 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, CA, 2003.

Automated Generation of Image Products for Mars Exploration Rover Mission Tactical Operations

Doug Alexander
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Douglass.A.Alexander@jpl.nasa.gov

Payam Zamani, Robert Deen, Paul Andres, Helen Mortensen
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, USA
Payam.Zamani@jpl.nasa.gov

Abstract - *During the period of development prior to the January, 2004 landing of the Mars Exploration Rover (MER) project's twin robotic vehicles on Mars, mission operations personnel recognized the need for timely generation and delivery of camera image products for rover traverse planning purposes. The task was assigned to the Multimission Image Processing Laboratory (MIPL), an element of the Jet Propulsion Laboratory (JPL). This paper will report on the ensuing design that involved sequentially transporting, or "pipelining", telemetered MER camera image data between disparate computer processes executed in parallel across multiple machine resources. Discussion will touch on the fundamental aspects of the system's event-driven processing strategy that provided autonomy in its operation. Overviews of the interconnecting process streams will be provided. In the end, it will be apparent to the reader that MIPL designed a system of image product generating systems built with robustness to meet rover planning requirements and with sufficient versatility to meet expanding operations needs in short order.*

Keywords: MER, JPL, MIPL, Mars, rover, image, product, data, pipeline, process, work flow.

1 Introduction

In January of 2004, NASA's Mars Exploration Rover (MER) mission successfully landed the "Spirit" and "Opportunity" rovers on the Mars surface. The techniques involved with remotely operating these mobile vehicles on a distant planet were highly dependent on the ability of mission operations personnel to receive and analyze imaging data that was acquired by each rover's set of engineering and science camera instruments. Once the data were telemetered to the Ground Data System (GDS) at the Jet Propulsion Laboratory (JPL), they had to be processed in such a way so as to optimize the extraction of navigational information embedded in the images, and with such timeliness that the arduous efforts inherent with analyzing the data and planning same day rover commanding were minimized.

This paper presents a discussion of an automated end-to-end system of data product generating systems designed to accomodate the *in situ* nature of MER rover operations. Developed by JPL's Multimission Image Processing Laboratory (MIPL), the system involved an integration of software programs that processed rover camera instrument data into a variety of unique image data products critical for rover operations traverse planning. Henceforth termed the "Pipeline" for convenience in this paper, the system was equally adept at processing non-image science instrument data for analysis by the science instrument teams. The system's name alludes to the notion that the digital data was sequentially transported, or "pipelined", from one component product generating system to the next. The fundamentals of the Pipeline's event-driven architecture and how they allowed for nearly complete autonomy of the Pipeline operation will be discussed.

The Pipeline's resultant data products were many and their descriptions are extensive. Discussion will touch lightly on the application software developed by MIPL for each data product. Detailing the characteristics of each product type is left as a topic for another paper [1], and instead focus will be placed on discussing the interdependencies between the element application processes as they are laced in the Pipeline's framework.

2 Overview

The basic objective of the Pipeline was to process telemetered camera instrument data into image data products, convert them into terrain maps, and subsequently complete their delivery onto the GDS. The time frame for product delivery had to be short enough to sufficiently allow for planning and uplinking of rover maneuver commands by rover operations short-term planners, the primary customers. Processing within this "tactical" timeline, measured on the order of hours, satisfied the requirements of two other types of operational customers: a) science planners, who were tasked with targeting features of interest found in the images for incorporation into short-term rover traverse plans, and b) mobility analysts, who were responsible for reviewing image data to determine where the rover actually had moved in comparison to the nominal traverse plan for the previous day. A fourth customer, the long-term planners who analyzed multi-image mosaics to plot the course of rover movement several days in advance, operated within a more casual "strategic" timeline measured in days.

Delivery of data products to operational users of the GDS inside JPL's secured flight local area network (LAN) was facilitated by a file server called the Operations Storage Server (OSS). Configured as an immense directory structure hierarchy, the OSS supplanted a customer database on the GDS. Outside the LAN, where dispersed elements of the science instrument teams awaited image data while residing at home institutions, delivery was made using a system called FEI designed at MIPL to provide secure data transfer across the network. See Section 5.5.3 for more discussion of this system.

3 System Environment

In the MER GDS configuration, the Pipeline was tightly choreographed with a set of upstream processes managed by JPL's System Software (SSW) team and various downstream customer entities. See Figure 1 for a high level diagram of the Pipeline's placement within the context of the MER GDS.

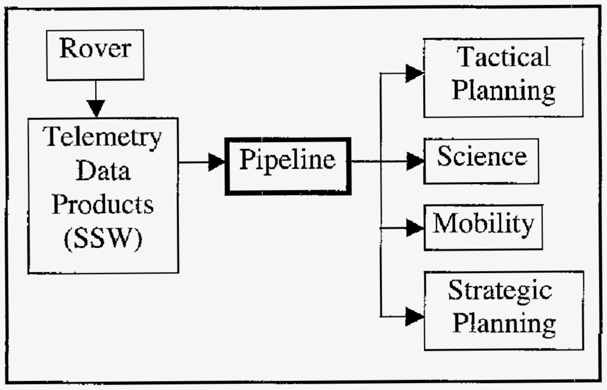

Figure 1. Data Flow in the GDS

3.1 System Hardware

The MER GDS hardware architecture utilized four redundant Sun file servers (NFS) to service a number of Sun/Solaris or Intel-Linux workstations. At the time of initial design, the target platform on the MER GDS for the Pipeline system was unknown. Ultimately, the final Pipeline computing engine was comprised of four dual-processor Intel-Linux workstations per rover mission, each having 1GB of RAM while running at clock speeds of 2.5Mhz.

3.2 Application Software

The Pipeline wouldn't have anything to do if it weren't for the applications that it managed. These were the programs that processed the data into a variety of unique product types. They were in essence the systems that the Pipeline integrated. The Pipeline's architecture allowed application programs to "plug in" with relative ease and minimal configuration. There were several main classes of applications: a) telemetry processing, b) derived image production, c) terrain generation, d) format conversion, e) data delivery, and f) image display.

In addition to the core application programs developed by MIPL, two external programs had to be integrated to support generation of products for the Mini-TES instrument and 3-dimensional terrain meshes.

3.2.1 First Order Products

The MIPL application software supporting the MER project drew a large portion of it's heritage from the Mars Pathfinder (MPF) and Mars Polar Lander (MPL) projects. The telemetry processor ("telemproc") was responsible for digesting raw telemetry data into first order science and operational data products, called Experiment Data Records (EDRs). The telemproc for the all rover engineering and science instruments, with the exception of the Mini-TES instrument, was developed in-house at MIPL and was a direct descendant of the Polar Lander's telemetry processor. The Mini-TES telemproc was developed at Arizona State University.

As shown in Figure 1, the Pipeline processing of EDRs began at the point of interface with SSW processes, which reconstructed the packetized rover instrument telemetry data resident on JPL's Telemetry Data Subsystem (TDS) into data product (DP) file pairs. Comprised of a binary instrument data file and an associated metadata file, each DP was automatically delivered by the SSW processes into an OSS directory called the DP Queue, where they were gathered by the Pipeline for immediate ingestion by the appropriate telemproc.

3.2.2 Derived Products

There were as many as 18 derived image products, called Reduced Data Records (RDR's), generated for each original EDR. A full accounting of each product type is provided elsewhere [1], but the suite included product applications such as radiometric correction, stereo correlation and XYZ generation [2], range (distance) information, robotic arm reachability [3], terrain slope information, and a variety of multi-image mosaic map projections. There were 14 applications written using the "VICAR" image processing system, all based on a common library (Planetary Image Geometry, or PIG) which handled all mission-specific details [4]. They were largely inherited from previous missions such as MPF and MPL and will be further reused in the future Phoenix and Mars Science Laboratory (MSL) missions.

Terrain generation was handled by SUMMITT, a set of terrain building software developed at JPL outside of MIPL. These applications converted the raw XYZ values into a unified terrain mesh used by rover planners for traverse navigation [5].

Figure 2 illustrates the data flow between the various RDR generating processes, starting with the image EDRs.

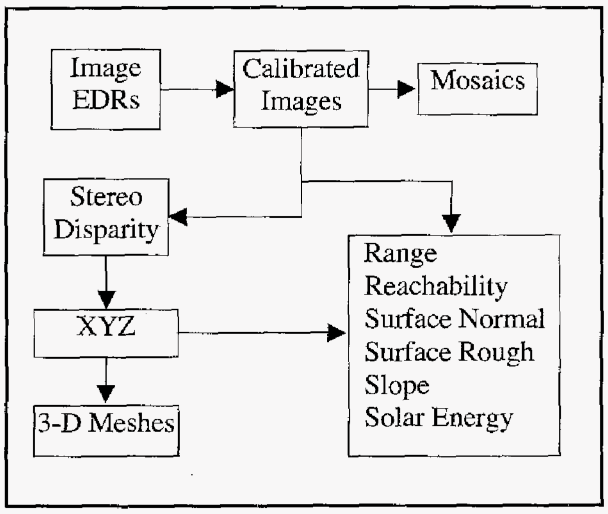

Figure 2. Simplified RDR Application Data Flow

The format conversion application was written in Java using the Image I/O mechanism. It converted any supported format to any other, but was primarily used to convert VICAR-format imagery coming from the RDR generation programs to the standard Planetary Data System (PDS) format required by MER. The important point is that it preserved metadata during the conversion process. It was also used to make JPEG's of the EDR's for public distribution.

The aforementioned FEI data delivery system and an image display system were also developed at MIPL, but are generic services used by many missions. See Sections 5.5.3 and 5.5.4, respectively, for details.

Because all of these applications were developed at different sites for different reasons, they were not consistent in terms of calling sequences. Some took simple parameters on the command line, others required input files be constructed. Some required a single image input, some required a stereo pair, and some required a whole collection of inputs. Logging messages were printed and formatted very differently. Most troublesome, the success/fail status returned by the applications were all different. Handling these inconsistencies in the Pipeline turned out to be one of the most challenging aspects of its development.

4 Performance Requirements

There were three major timing requirements levied by the MER project on MIPL for operations and science product generation:

- EDR products had to be generated and saved onto the OSS file server within 60 seconds of their arrival on the ground. Actual performance varied from 6 to 12 seconds depending on the instrument and size of data product.
- RDR products, with the exception of the terrain mesh, had to be produced within 30 minutes after the end of a telemetry downlink session. During the MER extended mission, new RDRs were introduced, such as solar energy and slope maps [3], and were exempt from meeting this requirement.
- Generation of the 3-D terrain meshes had to completed within one hour from the end of the downlink window. Actual performance varied and occasionally, depending on the size and number of meshes, this requirement was not met for the final mesh. Generation of this product required manual initiation since there was no automated method for broadcasting an end-of-downlink event.

It should be noted that most of the bottleneck in processing was due to the application programs as opposed to the "glueware". Still, the requirements had to be addressed at the time of the Pipeline design so as not to add to the latency already experienced at the application processing level.

In addition to the timing requirements, there were other requirements imposed by MIPL developers for robustness:

- Distribute all products to the MIPL catalog residing outside the flight LAN within 10 seconds of their creation.
- Allow for multiple Pipelines to run in parallel, independent of one another.
- Provide ability to manually reconfigure process loading across multiple workstations. Nominally, four machines were used to support each rover mission's data processing.
- Provide ability to halt or resume execution of the Pipeline at any point in the processing.
- Provide ability to log processing history and have real time tracking of products.
- Provide ability to perform special Pipeline processing "privately" in user directories away from the nominal OSS hierarchy.

5 Design and Implementation

In simple terms, the Pipeline was developed as a single parameterized Bourne shell script that, once initiated by command line at the shell prompt, spawned numerous child process streams across GDS machine resources at the control of the user. Each stream was a serial sequence of specific processes serving a variety of purposes, such as invocation of application software, PDS labeling of data products, and delivery of products to specific directories on the OSS by Sol and instrument type, as well as to external customers outside the LAN.

5.1 Programming Language

Selection of the programming language for the Pipeline development was predicated on a few key factors over a year before the MER mission's landing of the rovers in January of 2004. At that time, the MER project had yet to determine the type of hardware to be used for the GDS, and this prohibited MIPL developers from confidently knowing which versions of various software would be available. While the hardware resources were as yet unknown, it was established that the MER GDS would provide for a Unix-based environment. Since Unix is prevalent on a wide range of computing systems, the probability was high that the typical system user would have some level of Unix experience. The power inherent in the Unix language combined with user familiarity became a prime reason for MIPL developers to build the Pipeline as a Unix shell script. The selection was validated when the MER project settled on Sun and Linux machines as the GDS hardware of choice, with Unix running on both the Solaris and Red Hat operating systems (O/S's), respectively.

The important point is that Unix shell programming languages are known entities and by scripting the Pipeline under Unix shell, the groundwork was laid for easy development of software tools that could supplement or hook into the Pipeline during MER mission operations. And since Unix shells ran on both the Solaris and Linux O/S's of the GDS, deployment of the Pipeline was expanded to multiple user environments.

The Bourne shell was chosen over Perl for the following reasons: 1) the version of Perl available on the GDS was incompatible with the version compiled on the MIPL development system, 2) it was felt that scripting in Bourne shell maintained the largest common denominator across the collective knowledge base of the Pipeline developers and operators, and 3) heritage from MPL, wherein the data product generation system was developed under the Bourne-again shell (Bash). It's not to say that selecting Perl as the programming language wouldn't have had its merits as well.

5.2 Constraints

MER project policies governing the GDS constrained the Pipeline in two areas of development. One issue was the regulation that no Data Base Management System such as Sybase, PostgreSQL or MySQL be allowed in the critical path of MER mission operations, and the Pipeline was part of that critical path. So instead of designing a system that utilized a database for operational functions such as auto-triggering of file I/O between processes, an area of design very familiar to MIPL developers as demonstrated during past missions such as MPF and MPL, an alternative strategy had to be adopted. The second issue was a project policy that essentially prevented Pipeline access to a Web server inside the flight LAN, which restrained the distribution of the data product tracking capability.

The resolutions to these issues within the Pipeline design are discussed in subsequent sections in this paper.

5.3 Fundamental Strategy

The fundamental attribute of the design was the ability for each process within a stream to be event-driven. This was exemplified in two general forms within the Pipeline: 1) testing file residence in temporary OSS directories, and 2) testing file attributes by application criteria. Returned status of file residence and criteria testing drove automatic selection of subsequent actions (ie., events) regarding the file's handling, and demonstrated event-driven processing at the lowest level of the Pipeline design.

5.3.1 File Residence Testing

Regarding the first form, in lieu of a relational database's event-triggered capability, each process had built into it a series of endless loops specifically calling the Unix programs "ls" and "find" to search directories on the OSS for files. The temporary directories essentially served as queues that harbored the data for subsequent searching, or polling, by other Pipeline processes. There were up to eight types of directory queues:

- Input Queue - Where all pending input files for a particular application program were stored.
- Input Buffer - An intermediate holding bin where the sets of input files unique by SCLK were moved one at a time from the Input Queue for immediate ingestion by the application program.
- Output Buffer - An intermediate holding bin that received the single set of data processed by the application program for subsequent actions by the Pipeline based on file attributes.
- PDS Queue - Received all data from the Output Buffer destined for PDS labeling.
- PDS Buffer - An intermediate holding bin that received one set of data at a time for immediate ingestion by the PDS labeling system.
- Output Queue – Where the final PDS-labeled versions of data products were received, either from the PDS labeling system or from the Output Buffer, depending on the data product.
- FEI Queue – Where data products destined for external delivery outside the LAN were linked.
- JEDI Queue – Where image EDRs destined for image display were linked.

Additionally, there were other temporary directories that supported contingency processing in the case that data products were not generated, or had to be regenerated: 1) a directory for backup of each data product's input file set, and, 2) directories for file links in the case of failed processing.

5.3.2 File Attribute Testing

Regarding the second form, in the cases of "found" files, their characteristics were tested against criteria for acceptance by the application process that resulted in one of two status types: "success" or "failure".

An example was the need for the stereo correlation process to ingest a pair of images, one acquired using the left camera and the other acquired using the right camera. So for any found image, criteria was designed to test for that image's matching partner, and processing of the image would not proceed until its partner image was found.

5.4 File Softlinking

Because of the breadth of the OSS directory structure and the product delivery requirements imposed on the Pipeline, file manipulation had to be quick and efficient. This was achieved in the Unix environment by using programs "ln -s" and "cp -s" to softlink data files from directory to directory, minimizing the amount of file copying. Also, file softlinking avoided problems with accessing partially-written files, since the O/S provided for softlink creation to be an atomic process.

5.5 Parallelized Approach

Satisfying the data product delivery requirements necessitated a parallelized stream approach, implicit with the concurrent spawning of multiple child process streams at the outset of the Pipeline's invocation.

The parallelism was at the level of product and process types, and was a function of the number of child streams that could be engineered, with degrees of performance realized through user-controlled distribution of streams across available machine resources. A total of five such streams were identified. Not all product types used all streams, but most used at least three : 1) application stream, 2) PDS labeling stream, and 3) product delivery stream.

5.5.1 Application Stream

This process stream housed the command line call to the application program, and endlessly polled the Input Queue directory for any and all qualified input files. The stream would move a single set of input files unique by SCLK into the Input Buffer directory, from where the application program ingested the data. Upon completion of the processing, the stream deposited the resultant data product into the Output Buffer directory. More fine-grained parallelism could have been had with multiple application streams invoked for the same data product type, and will be a topic for the future.

5.5.2 PDS Labeling Stream

The labeling stream was responsible for calling a Java transcoder program to extract a file's metadata and generate a file-appended label that was compliant to PDS standards. The stream endlessly polled the PDS Queue directory for the candidate files, and moved them one at a time into the PDS Buffer for immediate ingestion by the Java transcoder. Upon completion of the processing, the labeled data product was placed into the Output Queue directory.

5.5.3 Product Delivery Streams (2)

For data product delivery onto the OSS, a stream was spawned to endlessly poll the Output Queue directory for the final PDS-labeled versions of the data products. The stream called the Unix program "mv" to reassign the address of a particular product's file pointer, so that each found file was effectively moved to the file server instead of copied. As part of the move of RDR products, the stream incremented the version number in the product's filename as necessary to avoid overwriting versions of the same product already resident on the OSS.

For data products destined for delivery outside the LAN, yet another process stream was launched to endlessly searched the FEI Queue directory. Found files were then ingested by the File Exchange Interface (FEI) system. FEI was developed as a client/server application to transport data from a data center to client sites, utilizing Kerberos authentication for security [7]. Using FEI programs, data products were copied from the LAN to an external server at MIPL for rerouting to other external client sites.

5.5.4 Image Display Stream

This stream endlessly polled the JEDI Queue directory for EDR image product softlinks. If found, the softlinked file was ingested by client software called the Java EDR Display Interface (JEDI) for image display onto a user-specified monitor. This stream was vital to quick visual quality checking of the image EDRs.

5.6 Error Handling and Messaging

As part of each stream, messages and returned error print statements were generated at both the application program level and the Pipeline glueware level into a single logging text file. There was some inconsistency in the manner by which the application programs returned low level error status, so the Pipeline was engineered to auto-categorize application error types into broad themes and issue additional messages to simplify interpretation.

The log file was set in auto-scroll mode on the workstation monitors for visual monitoring, but it's verbosity made for difficulties in readily interpreting the information in real time. Instead, the greater value found in the log file came with the fact that it provided a permanent record which was searchable at a more leisurely pace in times of anamoly investigations.

5.7 Extensibility

The Unix-based scripting approach to the Pipeline design provided for quick "plug-in" of new capabilities that became necessary due to growing requirements during mission operations.

One example of this Pipeline extensibility came during the extended mission, when several new image products were envisioned that would make operations easier. As Spirit climbed into the Columbia Hills and Opportunity descended into Endurance Crater, the long-term planners realized they needed to be able to visualize the local slope around each respective rover [3]. Power constraints and dusty solar panels led to a need for a product showing locations where solar energy would be maximized. Spirit's ailing right front wheel motivated a product showing climb/descent. All of these were easily implemented using combinations of existing or sightly modified applications.

As another example, the science team used a hybrid version of the Pipeline specially developed to create photometry cubes [6]. This entailed adding yet another incompatible type of application – in this case programs written in IDL (Interactive Data Language, from Research Systems) and very different parameters for many of the standard processing steps. These changes were also readily incorporated in a relatively short time.

In both cases, integration of the new capabilities into the Pipeline was fast and easy. Within the Pipeline script, all application processes were spawned by the same function, so it was simply a matter of adding the command line call to the new application program as a new module block in the code.

6 Operation

6.1 Startup/Shutdown

The Pipeline script was invoked in the Unix shell though command line execution by a single human operator. Processing behavior was controlled via specification of command line parameters.

Although the Pipeline was designed to run autonomously for long periods of time (many days), procedurally, there were advantages to managing the processing sessions in smaller increments. The resulting policy was startup/shutdown of the Pipeline once per Sol, which allowed the operator to maintain the size and number of the temporary working files and directories under limits where NFS performance became noticable.

6.2 Product Verification

6.2.1 GUI-based Product Tracking

Given the complexity of the Pipeline processing and the quantity of data passing through on a daily basis, the need for a means to visually track the progress of processing at the product unit level became apparent. A system called "Product Update Tracking Tool" (a.k.a. PUTT) was created that presented the Pipeline pilot with a web page which visually indicated (via color) the completion status of each image data product.

PUTT was implemented using a small C program and a Perl script outside the Pipeline script and allowed for quick assessment of each product's status by: a) denoting the completion state of each pertinent application program in a graphical spreadsheet (GREEN for success, RED for failure), b) in the cases of failures, isolating and extracting the error statements from the application process logs, and c) providing pop-up window viewability into the log snippets for that product.

As the Pipeline started processing a new EDR, the PUTT program created a small XML "token" file into a temporary Sol directory on the OSS. This file was uniquely named using Spacecraft Clock (SCLK) and Spacecraft Identifier (SCID). As the EDR matriculated through various derived product application processes in the Pipeline, the contents of the token file were updated with status information that included: a) the name of the application program, b) the return status of the program's execution, c) the time of program completion, and d) the name of the program's processing log file pertinent to the EDR.

The Perl script, running every few minutes, collected all unique SCLK tokens from the current Sol and created the web page representation of the status information. The token file contents were concatenated into a single XML file and then converted to HTML by passing the XML through an XSLT filter. If errors were indicated in the token file the script created a second web page containing the appropriate section of the processing log file. Because of project constraints limiting web servers on the flight LAN, the HTML files were copied to a remote web server and were then viewable via a web browser.

6.2.2 Text-based Product Tracking

The use of temporary directory queues to collect file softlinks allowed a simple tool to be written that provided insight into the processing. It was not a GUI representation, but textual, developed as a Perl script to count the number of files in each directory and report as a text listing every 10 seconds or so. Optional parameters were added to control the listings by Sol.

6.3 Private Pipeline Mode

Nominally, the Pipeline placed its output data products into the OSS directory structure, where customers would "shop" for standard data products in subdirectories named by Sol, instrument, and product type. However, there often arose the need to create non-standard data products for special purposes at the request of a customer. These non-standard products could not be copied into the OSS as it would affect all the other customers who were expecting standard data. Simple adjustment of the Product Delivery stream created a "private" mode of the Pipeline that could deliver products into user-specified directories without touching the OSS. This was also extremely useful for MIPL analysts to test new processing methods.

7 Conclusions

The real test of any system is how well it performs in an actual operational setting. The Pipeline has been used daily for over 400 Sols on two rovers, processing in excess of 80,500 EDR's and 750,000 RDR's as of this writing. While there have been anomalies, none have been serious, and we have met our requirements. The Pipeline has proved itself to be robust and flexible, adapting to a changing mission environment.

While the MIPL Pipeline will not change significantly at this point, it is expected that some derivative of it will be used in future missions. Topics to investigate in the future include more fine-grained parallelism, better product tracking, use of a database (possibly optional) to help manage processing queues, more sophisticated data-flow options, and a more modular, plugin-style approach to application integration.

Acknowledgements

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] R.G. Deen, D.A. Alexander, J.N. Maki, "Mars Image Products: Science Goes Operational", Proceedings of the 8th International Conference on Space Operations (SpaceOps), Montreal, Canada, 2004.

[2] R.G. Deen, J.J. Lorre, "Seeing in Three Dimensions: Correlation and Triangulation of Mars Exploration Rover Imagery", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[3] C. Leger, R.G. Deen, R.G. Bonitz, "Remote Image Analysis for Mars Exploration Rover Mobility and Manipulation Operations", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[4] R.G. Deen, "Cost Savings through Multimission Code Reuse for Mars Image Products", Proceedings of 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, CA, 2003.

[5] J.R. Wright, A. Trebi-Ollenu, J. Morrison, "Terrain Modeling for In-Situ Activity Planning and Rehearsal for the Mars Exploration Rovers", submitted to 2005 IEEE International Conf. on Systems, Man, and Cybernetics, Waikoloa, HI.

[6] J.M. Soderblom, J.F. Bell, R.E. Arvidson, J.R. Johnson, M.J. Johnson, F.P. Seelos, "Mars Exploration Rover Pancam Photometric Data QUBs: Definition and Example Uses", Eos Trans. AGU, Vol 85 No 47, 2004.

[7] T. Huang, "Component Architecture: The Software Architecture for Mission Requirements", Proceedings of 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, CA, 2003.